Amendments to the Drawings:

Add new Figure 12.

Attachment: New Figure 12

REMARKS/ARGUMENTS

Objections to the Specification

To begin, Examiner makes a point that describing an SPC code as a very simple convolutional code could be misleading. The reasoning behind the comment was that, for soft-in soft-out (SISO) decoding of an SPC code (as would be needed in the iterative decoding of an LDPC code) it is convenient to simply use forward-backward processing on a two-state trellis, with the (two) state metrics at the start of the trellis also being pre-loaded so that state 0 is guaranteed to win, and with the (two) state metrics at the end of the trellis being pre-loaded so that state 0 is guaranteed to win. That is, a SISO decoder implementation for an SPC code can very much "look like" a SISO decoder implementation for a convolutional code. Nonetheless, to avoid possible confusion, lines 30-32 on page 1 have been eliminated. Also, on page 4, at lines 27-29, the text ", but as mentioned earlier, these can be considered to be simple convolutional codes" has been changed to, " - in LDPC codes (low density parity check codes)".

Examiner makes a comment about the use of the term "shared" data elements, mentioning "unpunctured" vs. "punctured", and "multiple copies". To use a classic Turbo code as an example, the systematic bits are only transmitted once, but in iterative decoding, both constituent code SISO processings make use of the channel samples associated with the systematic bits - and in this sense there is very much a "sharing" between constituent codes. This is actually a key aspect of a Turbo code - the SISO decodings for the two (or more) convolutional codes are able to make use of the same single set of channel samples for the systematic bits. For instance, with a Turbo code having an overall rate of r=1/3, in decoding, it is like having two rate r=1/2 codes. Examiner's comments seem to suggest that in some cases "multiple copies" of certain bits would be transmitted, but it is not clear to Applicant why such an approach would be used. Any bit involved in two or more constituent codes (i.e., "shared', "overlapped", "interlocked" etc. between codes) need only be transmitted once, with the received channel sample associated with said bit being used in every SISO decoding associated with a constituent code in which said bit is involved.

At the bottom of page 3 of Examiner's comments, Examiner brings up the terms

"composite codes" and "concatenated codes", and puts forth the viewpoint that "concatenated codes" is the "normal term of art". It is not clear to Applicant, for instance, that the term "concatenated codes" adequately conveys an encompassing of low-density parity-check (LDPC) codes, and LDPC codes are an important class of iteratively decoded error-corrected codes made up of smaller codes put together into more powerful aggregate structures. For this reason, Applicant prefers the terminology "composite codes". This term is intended to encompass Turbo codes, serially-concatenated convolutional codes (SCCC), low-density parity-check (LDPC) codes, dual-injection double-interlock (DIDI) codes, and so forth.

At the top of page 4 of Examiner's comments, Examiner brings up the issue of puncturing as related to Turbo codes. To address this matter, reference will be made to the classic journal paper on Turbo codes, Claude Berrou et al.'s IEEE Transactions on Communications paper entitled, "Near Optimum Error Correcting Coding and Decoding: Turbo-Codes" (patent reference AH). Four pages of this paper have been included with this response for convenience. Examiner's attention is first directed to Figure 4 of the journal paper, which shows a "Basic turbo-encoder (rate 1/3)". The text associated with this figure is found at the start of section III of the journal paper. The text states, "... for an R=1/3 encoding or punctured for higher rates." That is, the "nominal" code rate of a Turbo code formed from two rate r=1/2 systematic convolutional codes is rate r=1/3, as shown in Figure 4 of the journal paper and described at the start of section III of said journal paper. It is understood that there is no need to transmit multiple copies of any bits. Puncturing can then be used to achieve code rates higher than the "nominal" code rate. Applicant would prefer to leave the specification text relating to Turbo codes and puncturing as is: that is, a standard Turbo code is not considered to involve heavy puncturing, but rather "puncturing" is something that can be used to achieve code rates higher than the nominal code rate. Applicant would argue that this usage is consistent with the general literature on the subject matter.

To elaborate further on this matter, Applicant has included Diagram A with this response, which shows a Venn diagram depiction of the coded bits of the Turbo code shown in Figure 4 of Berrou's journal paper. Note that there are no multiple copies of any bits.

Examiner states at the top of page 4, "... Turbo codes (also referred to by applicant as 'PCCC codes')". Applicant notes that the "PCCC" term was not coined by Applicant, but rather is found in the literature. For example, please see page 2 of the non-patent reference included in Applicant's IDS entitled, "Serial Concatenation of Interleaved Codes: Performance Analysis, Design, and Iterative Decoding".

Continuing on with page 4 of Examiner's comments, Examiner brings up the matter of "interlocked (interleaved) weight", with the context being Turbo codes. To explore this matter, Applicant will again make reference to the Turbo code depicted in Figure 4 of Berrou's journal paper, as well as making reference to the associated Venn diagram depiction of Diagram A. The constituent codes of a Turbo code are convolutional codes, and when examining the distance properties of a convolutional code, one examines "error-events"; an error event is a departure from the all-zero state, up to the first subsequent return to the all-zero state. We are dealing with linear codes, and that is why one can use the all-zero state as a reference. Note that "error-event" analysis is by no means Applicant's terminology, but rather is prior art that can be found in any textbook covering convolutional coding. Consider any arbitrary error event of constituent code 1 (RSC code C1 in Figure 4 of Berrou's paper). Such an error event has a certain weight (the number of non-zero data elements - here, the number of bits that have value '1'). Now, with reference to the Venn diagram of Diagram A, some of the '1' bits of said event will be part of the "overlap" region (labeled X), and the remainder will be part of the region labeled Y1. The interlocked (interleaved) weight of said event is therefore the count of the number of '1' bits of said error event that belong to the region labeled X - that is, they are part of the "interlocked" (or, interleaved, etc.) region. In brief, Applicant's terminology "interlocked (interleaved) weight" merely refers to the portion of the "error event weight" (standard terminology in prior art) that happens to be part of the "overlap" region as depicted in the Venn diagram of Diagram A.

At the bottom of page 4 of Examiner's comments, Examiner makes the point that, "the first constituent code of an SCCC may be the same as the first constituent code of a PCCC", and Applicant completely agrees with this statement. How the first constituent code figures into the overall composite code structure, however, is entirely different for an SCCC code as compared to a PCCC code. To illustrate this, Applicant has included Diagram B with this response.

Diagram B shows a Venn diagram depiction of an SCCC code. Note again that there are no multiple copies of any bits - the number of bits involved in the second constituent code is precisely the same as the number of coded bits of the overall SCCC code. From the diagram, it can be seen that there is a key difference between an SCCC code and a PCCC code in terms of how the first constituent code is used within the overall composite code structure. Since all bits involved in the first constituent code of an SCCC code are interleaved (or in other words, "overlapped", or "shared", etc. with the second constituent code), then necessarily more than one bit per state transition interval of the first constituent code is being interleaved (or in other words, "overlapped", or "shared", etc.). That is, the distinction between an SCCC code and a PCCC code is not in terms of the first constituent code viewed in isolation, but rather in terms of how the first constituent code figures into the overall composite code structure.

To help clarify Applicant's use of the terms "interleaved", "interlocked", and so forth, reference will be made again to the Venn diagram of Diagram B. The region of intersection between the two circles Applicant refers to by any of the following terms: "overlapped bits", "shared bits", "interleaved bits", "interlocked bits". In this case, because the Venn diagram is for an SCCC composite code, the set of "interlocked bits" is precisely the same as the set of coded bits involved in the first constituent code. Note that Applicant's use of the terms "interleaved bits", "interlocked bits", and so forth is consistent between the specification's discussion of Turbo codes and the discussion of SCCC codes, and remains consistent throughout the specification: such bits correspond to a region of intersection between constituent codes in a Venn diagram depiction of a composite code.

On page 5 of Examiner's comments, Examiner brings up the "interleaved/interlocked" issue as connected with H-TCC codes. Applicant hopes that the discussions provided above on this terminology will aid in the understanding of the use of this terminology in this context. However, H-TCC codes have been deleted from the specification.

Continuing to the next paragraph on page 5 of Examiner's comments, Examiner brings up Figure 5, which depicts the code structure of an example dual-injection double-interlock (DIDI) composite code. To address Examiner's concerns, Applicant has included Diagram C, which

shows a Venn diagram for the example DIDI code of Figure 5. Note that the two circles of the Venn diagram are exactly the same. This is not a mistake. The set of coded bits involved in the first constituent code is precisely the same as the set of coded bits involved in the second constituent code (again, for the example DIDI code of Figure 5). Note that Applicant is careful to use the word "set" here, and not "sequence" - the bits do not appear in the same order in the two constituent codes, and so the word "sequence" is not appropriate. A DIDI code cannot be encoded by first encoding constituent code 1, performing a re-ordering operation, and then encoding constituent code 2. Nor can a DIDI code be encoded by first encoding constituent code 2, performing a re-ordering operation, and then encoding constituent code 1. Both constituent codes must be taken into account *simultaneously* in the encoding of this type of composite code. This might indeed seem quite puzzling at first, as this situation is markedly different from that of Turbo code encoding or SCCC encoding . A general approach to DIDI encoding will now be outlined. The first constituent injection code amounts to a set of linear constraints (i.e., linear equations) that the coded bits must satisfy. Similarly, the second constituent injection code amounts to another set of linear constraints (i.e., linear equations) that the coded bits must also satisfy. Thus, the overall composite code can be considered to be represented by a (large) set of linear equations. Some of the coded bits are simply information bits (i.e., user data bits), and so the values of these bits are known at the outset of encoding. The remainder of the coded bits are unknown variables, whose values can be determined using the set of linear equations that represents the overall composite code. Thus, what we have is a linear algebra problem and the problem may be solved just like a normal simultaneous system of equations in linear algebra. Starting with a matrix representing the constraints of the DIDI code (parity-check matrix **H**), one can perform row-reduction so that each unknown variable is expressed as a function of known variables only (i.e., the information bits). This results in a generator matrix **G** that expresses each unknown bit as a function of the known bits (i.e., the information bits). Note that the row-reduction can be done offline - there is no need to perform this operation for every block that is encoded. The patent specification goes into greater detail describing these matrix manipulations. The fundamental point is simply that DIDI code encoding *cannot* be achieved by encoding the first injection code and then the second, nor by encoding the second injection code and then the first - the two sets of constraints are *simultaneous*. Applicant notes that it is not obvious how to

explicitly illustrate DIDI encoding in a figure other than by showing a matrix multiplication operation, as in: $\mathbf{v} = \mathbf{Gu}$ (page 33 line 28), but this provides little insight. This is why Figure 5 is provided, along with Figures 6 and 7, with accompanying descriptive text on pages 33-38 specifically dealing with the topic of DIDI encoding. DIDI encoding is indeed altogether different from how either Turbo codes or SCCC codes are encoded. Nonetheless, just because some linear algebra is connected with DIDI encoding in no way makes these codes less powerful - in fact, it can be argued that such codes can be more powerful - they involve such complex inter-relations between coded bits that simultaneous systems of equations are connected with their encoding.

In the same paragraph from page 5 of Examiner's comments, Examiner suggests replacing "re-ordered" with "re-ordered and then further encoded", referring to, for instance, page 8 line 8 of the specification. Applicant notes, however, that the specification is carefully worded to express: composite code encoding that produces a result that simultaneously satisfies two constraint sets; the specification intentionally does not describe: a serial-style encoding, where first an injection constituent code is ~~encoding~~ encoded, re-ordering is performed, and then some other code is encoded. In many cases, a serial-style encoding may not be possible, as described in the previous paragraph, and to encode the composite code, one may need to apply linear algebra. In some cases, serial-style encoding may be possible, but the intent of the specification text in question is to describe any composite code made up of an injection code and some other code, irrespective of whether or not such a composite code can be encoded in a serial-style fashion.

It is interesting to note that Examiner is in fact touching on why it could be misleading to refer to DIDI codes as "concatenated" codes - because they cannot be treated as an encoding of one code, followed by an encoding of another code (the classic Forney-style concatenated coding model, with an "outer" code and an "inner" code - e.g., concatenating a convolutional code with a Reed-Solomon code).

The Examiner's suggestions concerning the description of Figure 3 have been adopted.

On page 6 of Examiner's comments, Examiner states "However, as the rate of the DIDI

code example is given as 3/4 for the concatenation of two rate-7/8 injection codes...". Applicant notes, however, that a DIDI code is not a (Forney-style) concatenation of two injection codes (i.e., with an "outer" code and an "inner" code), as it would seem Examiner may be suggesting. Rather, a DIDI code is a composite code formed from two injection codes as described in some length above. In the example DIDI code, the first constituent injection code forces constraints on the coded bits that require 1 bit for every 8 coded bits in order that said constraints can be satisfied. Similarly, the second constituent injection code forces constraints on the coded bits that require 1 bit for every 8 coded bits in order that these additional constraints can also be satisfied. Thus, for the overall composite DIDI code, 2 out of every 8 coded bits are needed to simultaneously satisfy the constraints imposed by both constituent injection codes. This leaves 6 out of every 8 coded bits as information bits. Thus, the overall code rate of the example DIDI composite code is (8-1-1)/8 = 6/8, or r=3/4.

At the bottom of page 6 of Examiner's comments, Examiner brings up the "parity check matrix for the overall composite code" (page 38 line 5), and suggests that this matrix implies some sort of "parallel concatenation" of injection codings (unclear to Applicant what this means exactly). Applicant notes, however, that the parity check matrix provided in the specification on page 38 at line 5 indicates that the coded bits of the overall composite DIDI code must simultaneously satisfy the constraints of the first constituent injection code *and* of the second constituent injection code (with of course there being a re-ordering between the two codes - hence the re-ordering matrix **R** appearing in the $\mathbf{H}_c$ matrix definition). This is consistent with the explanations that have been provided in the preceding paragraphs, and with the descriptions in the specification. A DIDI code is a composite code involving two constituent codes whose constraints must be simultaneously satisfied. A DIDI code cannot be viewed as a concatenated code, with an outer code and an inner code, nor as two codes that can be encoded in parallel and independently. The portion of the specification text that specifically describes DIDI encoding can be found starting at line 20 on page 33, and continuing through line 22 on page 38, with three figures being associated with this description of DIDI encoding: Figures 5, 6, and 7.

On page 7 of Examiner's comments, Examiner brings up various suggestions to improve the clarity of the specification. On page 3 at line 20 of the specification, "but with SCCC, all of

the coded data elements of one convolutional code are provided as data elements..." has been changed to, "but with SCCC, all of the (unpunctured) coded data elements of one convolutional code are provided as data elements", and the parenthetic phrase at the end of the sentence has been deleted. On page 5, line 29, reference to "the state sequencing state" has been changed to "a state". On page 5 at lines 32-33, "for state transition intervals with inserted data elements" has been deleted. On page 6, at lines 2-3, "state sequencer state" has been changed to "state". On page 6, at lines 30-31, "equal to" has been left in, because the primary coded data elements are indeed "equal to" the data organization output sequence. Examiner has suggested alternative wording for use on page 6, line 14, page 10, lines 9 to 10 and page 11, line 10. However, it is respectfully submitted that the current wording is clear. The idea is that the output sequence includes every data element of the sequence of source data elements and also includes on an ongoing basis inserted data elements. On page 8 at line 12, "consistent with the first code" has been changed to "consistent with the first code (i.e., another injection code, but not necessarily of exactly the same form as the first injection code)". On page 8 at line 33, "be" has been added, and "more generally" has been set off by commas. Page 12 lines 11-13 has been amended to refer to "exactly two constituent codes". On page 12 at line 19, the comma has been changed to "or". Examiner has requested that the expression "injection codes" be amended to refer to "injection code encoder" in a number of places. As discussed previously, the DIDI code is not realized by two separate encoders. Rather, an overall code is produced that satisfies the constraints of two separate codes. Because of this, it would not be accurate to refer to first and second constituent injection code encoders as this would imply two separate encoding processes. On page 32, the expression "both of the injection codes might be the injection code of Figure 1" has been amended to read "both the injection codes have a structure defined by the injection code encoder of Figure 1".

Continuing with the same paragraph of Examiner's comments, but now on page 8, Examiner suggests replacing "sequencing bit stream" with "systematic bits in the [first/second] bit stream". The term "systematic bits" means "information bits". Now, in a DIDI code, the sequencing bit stream (see Figure 1) of the first constituent injection code is, with re-ordering, precisely the sequencing bit stream of the second constituent injection code. It would be

incorrect to state that only the information bits (i.e., the systematic bits) are the same between the two streams. In a Turbo code, that is indeed the case - only the information bits are interleaved - but the structure of a DIDI code is entirely different from that of a Turbo code, with more than just information bits being shared between constituent codes.

On page 36, line 8 amendments have been made to refer to Figure 1 as an encoder. However, equivalently Figure 1 defines a code. The point is that once the structure of a code is defined, for example as shown in Figure 1, there are other ways of producing the same code, for example simply using a generator matrix G. However, depicting a code simply by a generator matrix would obscure the code's structure. A code having the code structure depicted in Figure 1 would not necessarily be processed by a circuit such as shown in Figure 1, but by any circuit that produces the equivalent output. Finally, on page 33, line 6, Applicant would prefer to leave the language as it currently stands.

Objections to the Drawings

Examiner makes a valid observation that Figure 5 does not explicitly depict the encoding of a dual-injection double-interlock (DIDI) composite code. Figure 5 is merely intended to help convey the code structure of a DIDI composite code and in so doing implicitly provide insight into DIDI code encoding. The encoding of a DIDI code must simultaneously take into account the constraints imposed by both constituent injection codes, and for this reason, it is not evident how to come up with a figure that explicitly depicts the encoding. The encoding of a DIDI code cannot be depicted as two separate injection code encodings and an interleaving because of the necessity to simultaneously take into consideration both constraint sets. This is unlike the case with Turbo codes, for which the encoding can be easily depicted as, for example, two separate convolutional encodings and an interleaving. Using a dual-tailbiting two-constituent code Turbo code as an example, there is no need for either constituent convolutional encoding to "know about", or in any way to take into account, the other constituent convolutional encoding, and thus it is easy to depict such a Turbo code encoding in a figure (and indeed, one often sees figures showing "Turbo code encoding"). With a DIDI code, however, an analogous depiction is simply not possible. Applicant respectfully requests that Figure 5 be allowed to remain as is for the

reason that, even though the figure does not explicitly show the encoding of a DIDI composite code, the figure, by illustrating the code structure of a DIDI composite code, implicitly describes encoding. From Figure 5 (and Figure 1), one skilled in the art of error-correction coding could, for example, come up with a parity-check matrix **H**, row-reduce this matrix to result in a generator matrix **G**, and then use this generator matrix **G** to encode any arbitrary information sequence **u** to produce a code word **v**, as in $\mathbf{v} = \mathbf{Gu}$. To clarify this, references to Figure 5 in the description have been amended to refer to a code structure of a composite code featuring two injection codes whose primary coded data elements overlap completely.

Concerning the Examiner's request for a Figure showing a state sequencer with N = 8, 16, 32, it is noted that Figure 1 explicitly shows an 8-state sequencer (i.e., an N-state sequencer, with N=8). The same Figure applies for other values of N. For this reason, it is respectfully submitted that it is not necessary to reproduce Figure 1 for each value of N.

Concerning the claim 8 embodiment, Applicant submits herewith new Figure 12 that depicts the subject matter of that claim. A new paragraph has also been added to the description.

Claim Objections

In paragraph 5 of the detailed Action, Examiner has objected to claims 4, 5, 8-10, 14, 16-19, 20-24 and 40-42 as being of improper dependent form. In the amended claims being submitted herewith, claim 1 has been amended so as to define an encoder that produces coded data elements satisfying a set of constraints, the set of constraints being equivalent to those implemented by another encoder. With this amendment, subsequent claims that previously referred to a set of constraints equivalent to the encoder of claim 1 can now refer to the encoder of claim 1 and thereby be of proper dependent form. This solves the dependency problem for claims 4, 5 and 8-9. Claims 10 and 14 have been cancelled.

Claim 19 was put into independent form. Claims 16-18 and 20 have been cancelled.

Claim 21 and claim 22 have been put into independent form. Claims 23 and 24 have been cancelled. Finally, claims 40 and 42 have been put into independent form and claim 41 has

been cancelled. It is important to note that "set of constraints" is a common way of referring to a code. For example, while one may be define a convolutional code with delay taps and combiners, when the actual encoder is implemented it is not necessary that such delay taps or combiners be implemented, but rather, any structure can be implemented that imposes the same constraints that are imposed by the defined convolutional code. Thus, where the structure of the another encoder referred to in claim 1 includes elements such as state sequencers, state to data element converters and data organization components, it is possible to equivalently encode using, for example, a single generator matrix. Such a generator matrix will still impose the same set of constraints as the elements of the recited encoder.

Claim Rejections 35 USC § 112

In paragraph 7 of the detailed Action Examiner has rejected claims 1-14, 17-19, 21, 23, 25 and 42 under 35 USC 112, second paragraph.

In claim 1, Applicant has amended the claim to only refer to the first of the two options identified by Examiner. However, it is noted that the "option ii" clause identified by Examiner does not, in fact, relate to auxiliary data elements. Rather, it relates to the portion of the description entitled "Data Organization - Further Notes" as found on page 24, lines 12 to 29.

Regarding the rejection of claim 4, this claim has been amended to reflect the amendments made to claim 1. Also, the claim has been amended to clarify that the two sets of constraints are not satisfied in sequence, as suggested by Examiner, but rather are satisfied simultaneously. This is consistent with the description and the above detailed discussion. Similar amendments have been made to claim 5.

Regarding Examiner's objection to the wording of claim 8, Applicant notes that the alternate wording suggested by Examiner implies a serial-type encoding, which may be applicable in some special cases, but cannot be assumed in general (as discussed earlier in this response). For this reason, Applicant feels that the current wording of claim 8 is more appropriate than the suggested alternate wording.

Claim 9 has been re-worded to address the vagueness rejection raised by Examiner.

Regarding Examiner's objection to claims 12 and 13, Applicant has clarified above that the option "ii" recited in claim 1 does not in fact relate to the auxiliary coded data elements. In any case, the option "ii" has been deleted from claim 1.

Claim 18 has been cancelled rendering the rejection of that claim moot.

Regarding claim 19, the difference between this claim and claim 15 is that claim 15 recites an encoder comprising a number of specific components. Claim 19 recites an encoder that produces a sequence of primary coded data elements that satisfy a set of constraints equivalent to those of the encoder of claim 15. This does not require the implementation of the encoder of claim 15, but rather any encoder that implements the same set of constraints. Claim 19 has been amended to place it in independent form to deal with this issue.

In view of the foregoing, early favorable consideration of this application is earnestly solicited.

Respectfully submitted,
ANDREW HUNT

By _____
Allan Brett
Reg. No. 40,476
Tel.: (613) 232-2486 ext. 323

Date: January 25, 2005
RAB:rld:bbp